Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Künstliche Inteligenz

Eine natürliche Interaktionsschnittstelle zur Steuerung von humanoiden fußballspielenden Robotern

Lutz Freitag

Matrikelnummer: 4226598

lutz.freitag@fu-berlin.de

Betreuer:

Prof. Raúl Rojas,

Dipl. Daniel Seifert,

Dipl. Hamid Reza Mobalegh

Berlin, July 17, 2011

Lutz Freitag

Abstract

Natural interfaces are designated to easily generate inputs which are significantly harder to realize using classic input devices such as mice or keyboards. In this bachelor's thesis a natural interface will be developed based on the Microsoft Kinect sensor for the soccer playing robots of the Freie Universität Berlin. The robots will imitate as accurate as possible the poses and movements of a user. Motions like standing up will be implemented by tracking the gestures of the user. The interface is designed to be intuitively understandable.

Zusammenfassung

Mit natürlichen Schnittstellen lassen sich Eingaben für Programme realisieren, die mit klassischen Eingabegeräten wie Mäusen oder Tastaturen nicht oder nur sehr schwer umsetzbar sind. In dieser Arbeit wird eine solche Schnittstelle für die fußballspielenden Roboter der Freien Universität Berlin mit einem Microsoft Kinect-Sensor als Grundlage entwickelt. Die Bewegungen und Posen eines Nutzers sollen dabei so genau wie möglich von den Robotern nachempfunden werden, um diese dadurch zu steuern. Teile der Robotersteuerung wie das Aufstehen sollen auch durch Gestenerkennung am Nutzer umgesetzt werden. Bei der Umsetzung wurde darauf geachtet, dass die Schnittstelle intuitiv und leicht verständlich gehalten wird.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ahnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

17. Juli 2011

Lutz Freitag

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Die FUmanoids 1
1.2 Natürliche Schnittstellen 2
1.3 Aufgabenstellung 3

2 Verwandte Arbeiten **4**

3 Hard- und Software der Roboter **5**
3.1 Besonderheiten 5
3.2 Antrieb 5

4 Grundlagen **7**
4.1 Tiefensensor 7
4.2 Kinect 10
4.2.1 Technik 10
4.3 OpenNI 16
4.3.1 Architektur und Datenfluss 17
4.3.2 NITE 18

5 Steuerung durch Posen **19**
5.1 Winkel statt Positionen 19
5.2 Berechnung der Winkel der verschiedenen Gelenke 21
5.2.1 Mathematische Grundlagen 22
5.2.2 Modellierung 24
5.3 Probleme durch unterschiedliche Körpergeometrien 32

6 Steuerung durch Gesten **32**
6.1 Funktionsweise 33
6.2 Aufstehgeste 34
6.3 Schussgeste 35

7 Laufsteuerung **36**

8 Fazit **37**

9 Ausblick **38**

Abbildungsverzeichnis

1 FUmaoind von 2011 2
2 Anordnung der Servomotoren in den Robotern 6
3 Stereokameraaufbau 9
4 Stereo Halbbilder, Wilfried Wittkowsky 2004 10
5 Tiefenbildgenerierung mit strukturiertem Licht 12
6 Koordinatensystem des Kinect-Sensors 13
7 Kalibrierungspose (Psi-Pose) 15
8 Architektur des OpenNI Frameworks 18
9 3D Darstellung der Basispose 21
10 3D-Darstellung eines Nutzers 25
11 Winkel an Scharniergelenken 26
12 Winkel an der Schulter 28
13 Winkel Berechnung an den Beinen 30
14 Zustandsdiagramm Aufstehgeste 34
15 Zustandsdiagramm der Schussbewegung 35
16 Visualisierung der Laufsteuerung 37

1 Einleitung

Humanoide Roboter, die sich menschenähnlich verhalten, stellen eine große Herausforderung dar. Die Proportionen und Form der Roboter sind zwar humanoid, jedoch sind es die Bewegungen oft nicht. Die FUmanoids, die fußballspielenden Roboter der Freien Universität Berlin, sollen sich ähnlich zu Menschen bewegen und sich dabei möglichst stabil verhalten. Dafür werden Bewegungen auf den Robotern ausgeführt, die menschlichen Bewegungen nachempfunden sind.
In dieser Arbeit wird eine natürliche Schnittstelle zur Steuerung der Roboter entwickelt, mit der die Roboter direkt von Menschen gesteuert werden können. Dabei werden mögliche Techniken beschrieben, die für eine solche Schnittstelle nötig sind, Probleme bei der Umsetzung erläutert und Lösungen ausgearbeitet.

1.1 Die FUmanoids

Die Freie Universität Berlin nimmt mit den FUmanoids seit 2007 in der Kid-Size Liga des RoboCups teil. Dies ist ein Wettkampf, bei dem internationale Bildungseinrichtungen im Roboterfußball gegeneinander antreten. Die Motivation hinter dem 1995 gegründeten Wettkampf ist es, im Jahr 2050 den amtierenden Fußballweltmeister mit einer autonomen Robotermannschaft zu besiegen. Bis dahin ist es allerdings noch ein langer Weg. Der derzeitige Stand der Roboterforschung lässt zwar großes Potential erkennen, was zukünftige Entwicklungen angeht, jedoch sind viele Bereiche der Robotik lange noch nicht ausgereift. [18] [10]

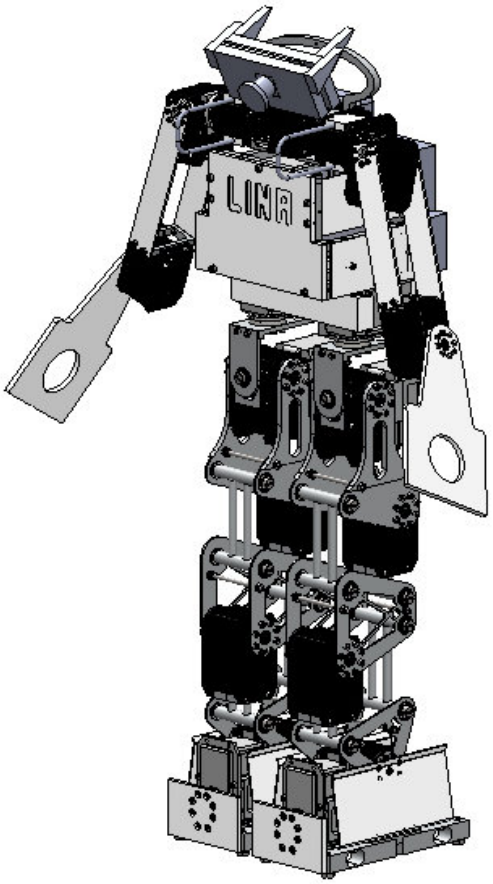


Abbildung 1: FUmaoind von 2011

In der KidSize Liga des RoboCups treten humanoide Roboter, das heißt Roboter mit zwei Armen, zwei Beinen, einem Torso und einem Kopf gegeneinander an. Die Roboter müssen dabei so gebaut sein, dass die Proportionen denen eines kleinen Kindes ähneln und sie dürfen nicht größer als $60cm$ sein.
Die FUmanoids nehmen regelmäßig am RoboCup teil und haben dabei beachtliche Erfolge erreicht. In den Jahren 2008 und 2009 wurden sie Vize-Weltmeister. [18]

1.2 Natürliche Schnittstellen

Im klassischen Sinn werden Computer durch spezielle Hardware bedient, die die Eingabefunktionalität übernehmen. Mit einer Tastatur können besonders schnell verschiedene Eingaben aus einem Alphabet erzeugt werden, wohingegen sich Mäuse eignen, um Eingaben zu erzeugen, die zu einem Ort gehören. Das Schreiben von Texten wäre ohne Tastatur fast nicht möglich und das

Bearbeiten von Bildern nicht ohne Maus. Beide Eingabegeräte haben ihre Stärken und Schwächen. In einigen Bereichen der Computernutzung haben sich Eingabegeräte etabliert, die für spezielle Anwendungen optimiert sind. Für Bildbearbeitungsprogramme eignen sich Grafiktabletts. Sie stellen einen Mausersatz dar, der in Form eines Stiftes und einer Unterlage realisiert wird. Die relative Position der Stiftspitze auf dem Tablett wird vom Computer in eine Mausposition umgesetzt und wenn der Druck des Stiftes auf der Oberfläche einen gewissen Schwellwert überschreitet, so wird das als Klick interpretiert.
Durch die Verbreitung von Touchscreens, wurden neue Eingabekonzepte möglich. Das Touchdisplay dient dabei sowohl als Tastatur als auch als Mausersatz. Die berührungsempfindliche Oberfläche gibt den Entwicklern und Designern von Benutzeroberflächen viele neue Möglichkeiten der Umsetzung von intuitiven Bedienkonzepten. [9]
Natürliche Schnittstellen stellen ein erweitertes Bedienkonzept dar. Eingaben können in Form von Gesten oder Bewegungen getätigt werden. Dabei gehen sie so weit, dass sie fast gänzlich ohne klassische Computerhardware auskommen. Der Nutzer kann ohne technische Hilfsmittel mit dem Computer interagieren. Er kann auf Elemente auf dem Bildschirm zeigen, sodass diese ausgewählt werden oder eine Greifbewegung machen, um Fenster zu ziehen. Die Bedienmöglichkeiten durch natürliche Schnittstellen sind dabei äußerst vielfältig. Grundsätzlich gilt aber, dass der Nutzer keine Hardware "in die Hand" nehmen muss, um den Computer zu bedienen. Es muss allerdings Hardware vorhanden sein, die den Nutzer und seine Eingaben registriert. Hierfür eignen sich Kameras oder Mikrofone.

1.3 Aufgabenstellung

Ziel dieser Arbeit ist es, eine natürliche Schnittstelle, also eine Schnittstelle, mit der man nicht direkt mit der Harware eines Computers interagiert, zu entwickeln, mit der die fußballspielenden Roboter der FU gesteuert werden können. Sie sollen sowohl direkt die Bewegungen des Nutzers umsetzen als auch durch Gesten gesteuert werden können. Dabei soll der Nutzer die gleichen Steuerungsmöglichkeiten haben wie die Roboter, wenn sie im autonomen Betrieb sind. Die Erzeugung der Steuerungsdaten soll auf einem dedizierten Computer geschehen und über WLAN an die Roboter übertragen werden.

2 Verwandte Arbeiten

Nach der Veröffentlichung der ersten offenen Treiber für den Microsoft Kinect-Sensor, entwickelte sich schnell eine große Gemeinschaft von Entwicklern, durch die viele interessante Projekte im Bereich natürlicher Schnittstellen entwickelt wurden. [7]
Einige Beispiele sollen hier genannt werden:

- Nao OpenNI
  Das Nao OpenNI Projekt stellt dem Nutzer eine zu dieser Arbeit ähnliche natürliche Schnittstelle zur Verfügung, um einen Roboter zu kontrollieren. Der Nutzer kann zwischen verschiedenen Modi wählen, um einen Nao-Roboter fernzusteuern. Man kann die Arme des Roboters direkt steuern, indem der Roboter die Armposen direkt umsetzt, das Laufen des Roboter, indem man im Laufmodus den eigenen Körpermittelpunkt relativ zu einem Nullpunkt bewegt oder auf einen Punkt zeigt, zu dem der Roboter gehen soll. Für letzteres muss der Roboter im Koordinatensystem der Kinect lokalisiert sein, also auch von der Kinect erfasst werden. Das Umschalten der verschiedenen Modi erfolgt durch spezielle Gesten mit den Armen.

- FAAST
  Das am kalifornischen Institute for Creative Technologies entwickelte Framework FAAST (Flexible Action and Articulated Skeleton Toolkit) stellt eine Umgebung bereit, mit der ein Nutzer durch bestimmte Bewegungen oder Haltungen Eingaben am Computer generieren kann. Das Framework kann dabei zwischen 26 verschiedenen Aktionen unterscheiden, die dann als Eingaben am Computer umgesetzt werden. Man kann beispielsweise die Cursorsteuerung dadurch realisieren, dass sich der Nutzer nach vorne bzw. hinten, rechts bzw. links neigt. [7]

- KinEmote
  KinEmote ist eine Steuerung für die Mediencentersoftware XBMC. Mit ihr kann ein Nutzer durch seine Medienbibliothek navigieren und Elemente Abspielen. KinEmote wird ausschließlich durch die Arme gesteuert. Bei der Eingabe wird zwischen drei Ebenen unterschieden. In der hinteren Ebene kann der Nutzer zum Hauptmenü zurückkehren und die Abspielfunktionen steuern, in der mittleren Ebene kann durch die Menüs navigiert werden und in der vorderen kann der Nutzer seine Auswahl bestätigen. [7]

3 Hard- und Software der Roboter

Humanoide Roboter brauchen zur Bewegung viele Freiheitsgrade, die sie von den Robotern der MidSize Liga deutlich unterscheiden. In der MidSize Liga haben die Roboter Räder, mit denen sie sich schnell auf dem Spielfeld bewegen können, ohne Gefahr zu laufen, dass sie umfallen.
Das humanoide Laufen, also das Laufen auf zwei Beinen, ist dagegen eine große Herausforderung. Anstelle von maximal vier Antriebsrädern werden bei den FUmanoids allein sieben Servomotoren pro Bein genutzt, um das Laufen zu ermöglichen. Insgesamt sind sowohl in den Robotern von 2009 als auch von 2011 der FUmanoids 21 Servomotoren verbaut. Dementsprechend sind die Bewegungsfreiheiten größer, aber der Aufwand der Stabilisierung wird mit der Anzahl der Freiheitsgrade komplexer.
Die aktuell genutzten Roboter der FUmanoids wurden im Rahmen der Diplomarbeiten von Jan Streckenbach und Moritz Fröhlich entworfen und im Juni 2011 fertiggestellt. Beim RoboCup 2011 in Istanbul haben die neuen Roboter den vierten Platz erreicht.

Details der Roboter von 2011: [18]

Höhe	$60cm$
Gewicht	ca. $4kg$
Servos	21 (3 pro Arm, 7 pro Bein, 1 im Kopf)
Prozessorboard	IGEPv2 DM3739 ($1GHz$ Cortex-A8 ARM)
Kamera	Logitech Quickcam 9000 Pro (VGA)
Netwerk	LAN, WLAN
Betriebssystem	Linux

3.1 Besonderheiten

Als Optik wird bei den Robotern eine 179° Fischoptiklinse genutzt. Damit sind die Roboter in der Lage einen Großteil des Spielfeldes zu sehen, ohne den Kopf bewegen zu müssen. Dies ist insbesondere dann nützlich, wenn mehrere Teile des Spielfeldes gleichzeitig beobachtet werden müssen. Positioniert sich der Roboter gerade zu einem Torschuss, so kann er gleichzeitig seine Füße, den Ball und das gegnerische Tor visuell erfassen.

3.2 Antrieb

Die Servomotoren, die in den FUmanoids verbaut sind, stammen von der Firma Robotis Inc. Sie zeichnen sich durch standardisierte Aufnahmen für

Achsen und Befestigungen aus und können mit einem seriellen Protokoll einzeln oder in Gruppen angesteuert und ausgelesen werden.
Die Servos unterstützen dabei folgende Grundfunktionen:

Beschreibung	Werte
Sollposition	0 .. 1023
Sollgeschwindigkeit	0 .. 1023
Torsionslimit	0 .. 1023

[16] [17]
Im Antriebsteil der Roboter befinden sich die stärkeren Servos (RX-64) und im Oberkörper die schwächeren (RX-28).

	RX-28	RX-64
maximales statisches Drehmoment	3,7 Nm	7,5 Nm
Gewicht	72 g	125 g
Getriebe	1 : 193	1 : 200
erreichbare Positionen	300°	300°
Winkelauflösung	0,35°	0,35°
maximale Winkelgeschwindigkeit	$0,6°/s$	$0,5°/s$

[16] [17]

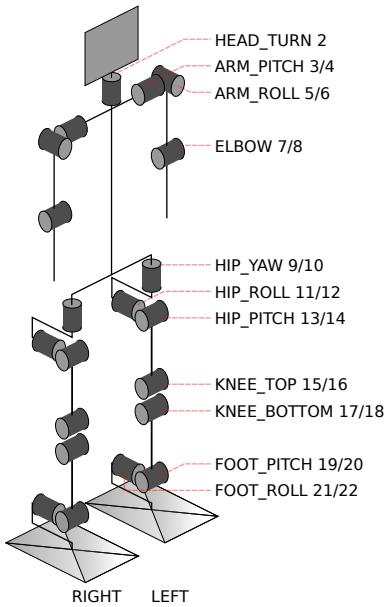


Abbildung 2: Anordnung der Servomotoren in den Robotern

4 Grundlagen

Im Folgenden werden Techniken zur Generierung von Tiefeninformationen erläutert.

4.1 Tiefensensor

Sensoren, mit denen die räumliche Entfernung von Objekten gemessen werden können, gibt es in vielen Ausführungen. Grundsätzlich gibt es zwei Klassen von denen einige Vertreter hier vorgestellt werden sollen:

- Aktive Tiefensensoren:

  Sensoren, die zur Messung der räumlichen Entfernung aktiv Messungen durchführen.

  - Laser-Abstandssensor

    Ein Laserstrahl wird auf das Objekt gestrahlt, dessen Entfernung gemessen werden soll. Aus der Laufzeit des Lichtes lässt sich die Entfernung bestimmen: $d = \frac{t_{Laufzeit}}{2v_{Lichtgeschwindigkeit}}$. Solche Sensoren sind vergleichsweise teuer und können nur einen Punkt gleichzeitig messen. Sind jedoch auch bei verschiedenen Oberflächen sehr genau und haben Reichweiten bis zu über 50 m. Im Aufbau als beweglicher Fächer lassen sich mit ihnen detaillierte Informationen über die Umgebung erzeugen.[19]

  - Infrarot-Abstandssensor

    Eine LED, die infrarotes Licht (nicht notwendigerweise infrarot, es hat sich aber als praktikabel etabliert) abstrahlt, beleuchtet das Objekt, dessen Entfernung gemessen werden soll. Aus der vom Objekt reflektierten Lichtintensität lassen sich Rückschlüsse ziehen, wie weit das Objekt entfernt ist. Solche Sensoren sind nicht teuer, dafür messen sie nicht gezielt. Befinden sich mehrere Objekte im Erfassungsbereich des Sensores, wird die Messung grundsätzlich verfälscht. Außerdem entstehen auch durch unterschiedliche Materialoberflächen oder -formen große Fehler. Ein kleines Objekt wird weniger wahrscheinlich gemessen als ein großes. Oberflächen, die infrarotes Licht wenig reflektieren, sind damit fast nicht zu

messen. Es gibt auch Ausführungen solcher Sensoren, die auf andere Wellenlängen des Lichtes zurückgreifen.

- Akustische Abstandssensoren

  Mit Laufzeitmessungen von Schallwellen lassen sich vergleichsweise gut Entfernungen messen. Solche Sensoren werden oft in Geräten genutzt, mit denen Räume vermessen werden. Akustische Abstandssensoren funktionieren, indem kurzzeitig ein hochfrequentes Schallsignal erzeugt wird, das von einer Oberfläche reflektiert wird und wieder zum Messgerät zurückkehrt. Aus der Laufzeit und der Schallgeschwindigkeit der Luft lässt sich nach $d = \frac{t_{Laufzeit}}{2v_{Schallgeschwindigkeit}}$ die Entfernung messen. Akustische Entfernungsmesser werden ungenauer, je schräger die Oberfläche ist, deren Entfernung gemessen werden soll, da hier weniger Schall reflektiert wird, der als Hintergrundrauschen interpretiert werden kann. Weiterhin verfälschen Luftfreuchtigkeit und Luftdruck das Ergebnis. Problematisch sind auch nicht-diffuse Echos, die jeweils als Nutzecho interpretiert werden können. Sind mehrere Objekte schräg hintereinander, sodass es eine sichtbare Linie zwischen dem Messgerät und den Objekten gibt, beispielsweise eine Reihe Säulen, so reflektiert jede der Säulen das Signal des Messgerätes. Je nach Bauart des Messgerätes muss dabei im Nachhinein eine Fehlerabschätzung geschehen.

- Aktive Triangulation mit einer Kamera und strukturiertem Licht

  Es wird strukturiertes Licht in die Szene projiziert, die wiederum von einer Kamera abgefilmt wird. Aus dem Verlauf des strukturierten Lichtes lassen sich Rückschlüsse auf die Oberflächenform ziehen. Da eine solche Technik in dieser Arbeit genutzt wurde, soll sie nachfolgend näher beschrieben werden.

- Passive Tiefensensoren:

  - Stereokamera

    Mit einem Aufbau von zwei Kameras, die die gleiche Szene beobachten lassen sich Informationen über die räumliche Beschaffenheit eben dieser erzeugen. Ein solcher Aufbau ist in Abbildung 3 skizziert. Inspiriert wurde diese Technik von der Natur, da auch die meisten

Lebewesen zwei Augen besitzen. Durch einen entsprechend dimensionierten bildverarbeitenden Apperat ist es dabei möglich, aus den zwei aufgenommenen Halbbildern räumliche Informationen über die Szene zu erhalten. Die eigentliche Schwierigkeit besteht hierbei darin, Bildteile (Features) aus dem einen Bild im anderen Bild wiederzufinden, zu matchen. Über den Versatz der Features in den Halbbildern und den Orientierungen der Sensoren lassen sich die räumliche Entfernungen berechnen. [4]

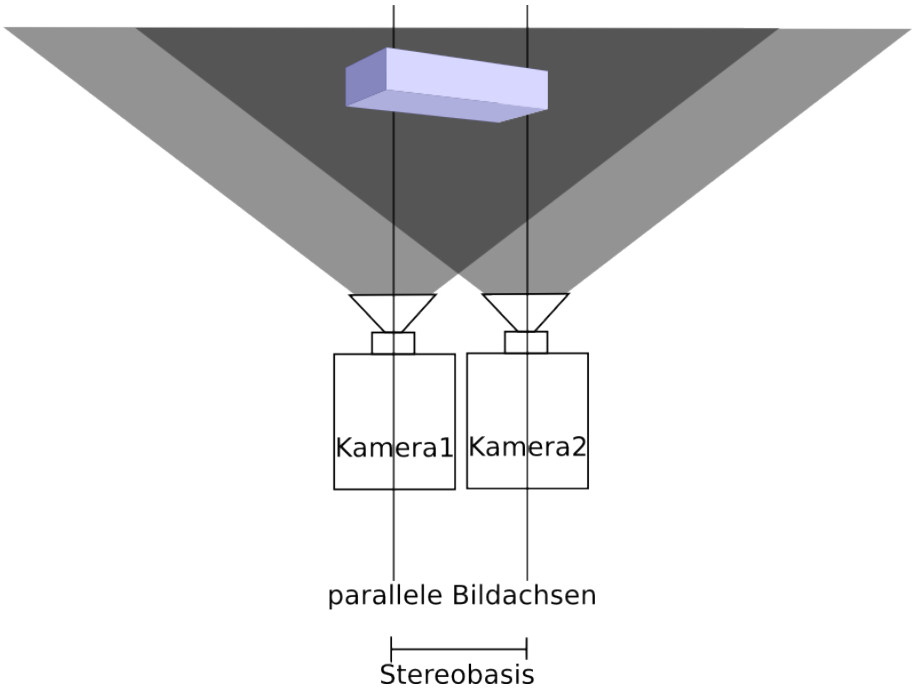


Abbildung 3: Stereokameraaufbau

Für eine akkurate Abschätzung der Entfernung eines Features muss nicht nur die horizontale Verschiebung berücksichtigt werden sondern auch die Skalierung der Features in den Bildern. Befindet sich ein Feature näher an einem Sensor, so ist es in dem Bild entsprechend größer. Gegenüber Rotationen muss das Matching nicht robust sein, wenn die Kameras nicht gegeneinander verdreht und ihre Bildachsen parallel sind. Man benötigt demnach einen Algorithmus, der Features in den Bildern findet, die skalierungsinvariant beschrieben werden können. [3]

Abbildung 4: Stereo Halbbilder, Wilfried Wittkowsky 2004

4.2 Kinect

Die Microsoft Kinect ist ein Sensor, der ursprünglich ausschließlich für die X-Box 360 entwickelt wurde. Er integriert eine Farbkamera, 4 Mikrofone und eine Tiefenkamera als Eingabesensoren. Die Tiefenkamera stellt dabei für die Entwicklung von natürlichen Schnittstellen die wichtigste Eigenschaft dar.[15]
Im Gegensatz zur Generierung von Eingaben, die auf Bildverarbeitung beruhen, beispielsweise durch eine Farbkamera, ist es durch die Nutzung von Tiefenkameras einfacher, Rückschlüsse über einen Nutzer zu ziehen und diesen zu erkennen.[5]
Bei der Nutzung des Sensors muss gewährleistet werden, dass der Sensor sich nicht im Raum bewegt. Ist dies nicht der Fall, so kann die Software, die das Tiefenbild auswertet, keine sinnvollen Ergebnisse erzielen.

4.2.1 Technik

Die Tiefenkamera der Kinect arbeitet mit strukturiertem infrarotem Licht, das seitlich vom Sensor ausgestrahlt wird. Das auf der Umgebung reflektierte Muster, wird dabei von einer monochromen Kamera aufgenommen. In der Kinect wird mit dem Light Coding Algorithmus das reflektierte Muster in ein Tiefenbild umgerechnet. Da das LightCoding Verfahren proprietär ist, kann keine Erklärung zum genauen Verfahren gemacht werden.[15]

Features:

- Farbkamera

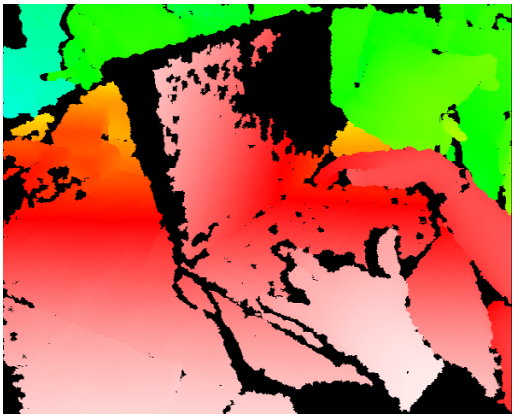
Auflösung	640 x 480 (VGA)
Farbtiefe	8 Bit (Beyer)
Öffnungswinkel	58° horizontal
	45° vertikal
	70° diagonal

- Tiefenkamera

Auflösung	640 x 480 (VGA)
Pixeltiefe	11 Bit (16 Bit, davon werden 5 nicht genutzt)
	Der Wert eines Tiefenpixels beschreibt
	die Entfernung zur x-y-Ebene des Sensors
Öffnungswinkel	58° horizontal
	45° vertikal
	70° diagonal

[15]

(a) Infrarotbild [1]



(b) Tiefenbild [2]

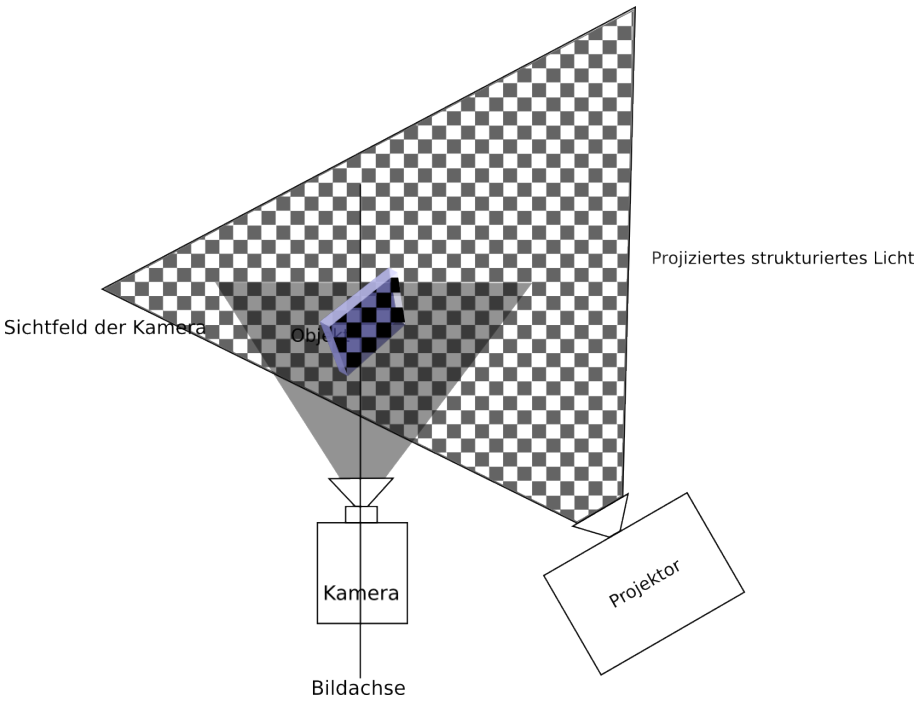


Abbildung 5: Tiefenbildgenerierung mit strukturiertem Licht

In Abbildung 5 wird beschrieben, wie Tiefeninformationen aus strukturiertem Licht gewonnen werden können. Das Muster, das vom Projektor rechts im Bild abgestrahlt wird, wird vom Objekt reflektiert. Durch die Verzerrung der Reflektion lassen sich Rückschlüsse auf die Orientierung des Objektes bzw. seiner Oberfläche gewinnen. Ist das strukturierte Licht außerhalb der Bildebene der Kamera gebündelt, kann außerdem durch die Abstände benachbarter Elemente des Musters auf die Entfernung des Objektes geschlossen werden. Ist ein Objekt näher, so werden die Abstände zwischen den Elementen im Bild größer. [4]

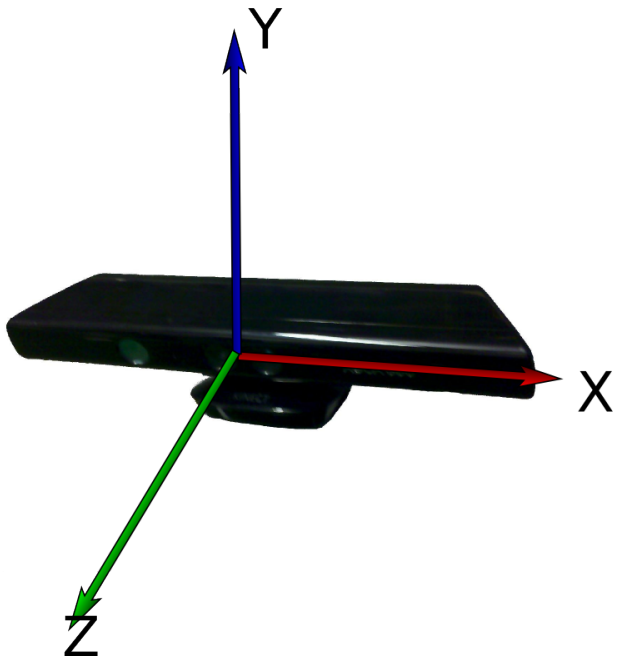


Abbildung 6: Koordinatensystem des Kinect-Sensors

Bei der Nutzung der Kinect muss beachtet werden, dass es mehrere Koordinatensysteme gibt. Einerseits steht in jedem Tiefenpixel die Entfernung des dazugehörigen Raumpunktes zu der Bildebene des Sensors, andererseits lassen sich auch alle Tiefenpixel mit den Bildkoordinaten in eine Punktwolke transformieren, in denen alle Koordinaten absolut sind. Der Ursprung des Koordinatensystem liegt dabei im Mittelpunkt des Sensors, siehe Abbildung 6 [15].
Durch die Nutzung eines Tiefensensors lassen sich viele Probleme weniger aufwändig lösen als bei der Nutzung von Farbkameras. Grundsätzlich ist es einfacher Objekte, bewegliche und feste, im Vordergrund vom Hintergrund zu trennen. Mit Tiefenkameras können Objekte auch direkt vermessen werden, ohne einen komplizierten Messaufbau herstellen zu müssen. Selbst wenn die Entfernung eines Objektes zur Kamera bekannt ist, ist die Berechnung der Geometrie des Objektes nur unter Annahmen machbar, die genaue Kenntnisse weiterer Objekteigenschaften voraussetzen. [3]
Im Folgenden werden exemplarische Verfahren vorgestellt, mit denen aus Tiefenbildern Informationen gewonnen werden können, die speziell für die Nutzung als natürliche Schnittstelle von Vorteil sind.

- Nutzererkennung
  Nutzer sind bewegliche Elemente in Tiefenbildern. Sie können analog zur Detektion von beweglichen Objekten in Farbbildern durch lokale

Veränderungen erkannt werden. Im Gegensatz zur Erkennung von Menschen in Bildern hat man jedoch durch den Einsatz einer Tiefenkamera genaue Möglichkeiten, die sich bewegende Punktmenge zu vermessen und deren Lokalisierung im Raum festzustellen. Eine geeignete Möglichkeit wäre, dass die Tiefenbilder über ein Zeitfenster gemittelt werden, um rauschen zu unterdrücken (Median-Mittelung). Die Hypothesen für einen Nutzer erhält man, indem man das aktuelle Frame mit den gemittelten Frames vergleicht. In einem Bereich, in dem der Unterschied besonders groß ist, kann sich ein Nutzer befinden [5]. Anschließend wird jede Hypothese vermessen und wenn sie innerhalb gewisser Dimensionen liegt, wurde wahrscheinlich ein Mensch gefunden. Weitere zusätzliche Tests, wie der Position der Hypothese zu ihrer Umgebung (Abstand zum Hintergrund, Orientierung zum Boden), ermöglichen die Unterdrückung weiterer falscher Vermutungen. Durch dieses Verfahren erhält man eine Maske, die so groß ist wie das Tiefenbild, wo in jedem Pixel die ID des zugehörigen Nutzers steht.

- Skelettierung
  Der Begriff Skelettierung wird hier für die Erkennung von Gliedmaßen genutzt. Es werden nicht die einzelnen Knochen erkannt, sondern Körperteile, die durch Gelenke miteinander verbunden sind. Wie z.B. Unter- und Oberarme.

  Um einen Nutzer und sein Skelett zu erkennen und zu tracken, muss der Nutzer kalibiert werden. Der Nutzer muss eine spezielle Pose einnehmen, bei der die Vermessung des Skelettes so eindeutig wie möglich umsetzbar ist. Dabei hat sich die sogenannte Psi-Pose (siehe Abbildung 7) etabliert.

  Die Skelettierung versucht nun ein Skelett in die Maske des Nutzer einzupassen. Hier kann von der Pose profitiert werden. Die drei höchsten Teile der Nutzermaske sind der linke Unterarm, der Kopf und der rechte Unterarm. Die Unterarme können vermessen werden, indem man sich an ihrem äußeren Rand so lange nach unten bewegt, bis der Punkt erreicht wurde, an dem man sich nur noch annähernd horizontal bewegen kann. Von diesem Punkt, bis zum obersten Punkt des Unterarmes verläuft der Unterarm im Skelett. Der Torso wird dadruch erkannt, dass er die größte zusammenhängende Punktmasse in der Maske ist. Die linke und rechte Begrenzung kann durch den ersten bzw. letzten Tiefenpixel ermittelt werden, der unterhalb des unteren Endes des nächsten Armes noch zum Nutzer gehört. Auf diese Art könnnen auch die Orte der Schultern erkannt werden. Die Höhe des Torsos kann analog

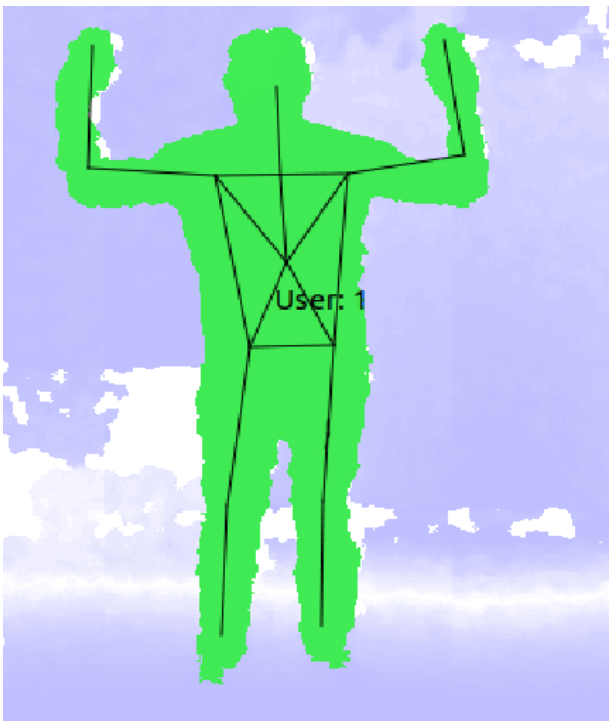


Abbildung 7: Kalibrierungspose (Psi-Pose)

durch den ersten Pixel der Nutzermaske ermittelt werden, der rechts oder links neben dem Kopf unterhalb desselben ist. Eine Alternative ist, dass der Torso genau so hoch ist, wie die Oberarme. Der Kopf wird dadurch erkannt, dass er der höchste Teil der Nutzermaske zwischen den Unterarmen ist. Die Verbindung zwischen Kopf und Torso wird durch den höchsten Punkt des Kopfes und dem Punkt zwischen den Oberarmen realisiert. Die untere Begrenzung des Torsos ist der höchste Punkt der Nutzermaske, der zwischen den Beinen ist, also der, wo die Nutzermaske eine Lücke aufweist. In der Kalibrierungspose sind die Beine gestreckt, was die Skelettierung erschwert, da man keine genauen Informationen über die Position der Knie erlangen kann. Hierfür eignet es sich, anzunehmen, dass sich das Knie auf der halben Strecke zwischen Torsoende und Füßen befindet. Bewegt der Nutzer seine Beine, kann die relative Position der Knie nachträglich justiert werden, indem für die Ober- und Unterschenkel lineare Funktionen angenommen werden, die sich in den Knien schneiden.

4.3 OpenNI

Im Zuge der Entwicklung und stetig wachsenden Popularität des Kinect-Sensors, gründete sich im November 2010 die Non-Profit-Organisation Natural Interaction, die sich derzeit aus PrimeSense, Willow Garage, Side Kick und Asus zusammensetzt. Die Organisation hat sich mit dem Ziel gegründet ein einheitliches Framework zu schaffen, welches die Datenverarbeitung von diversen Sensoren und durch vielfältige Datenaufbereitungsmodule vereinfacht [11].

OpenNI stellt ein Framework zur Verfügung, mit der ein Entwickler Funktionen bereitgestellt bekommt, mit denen sich natürliche Schnittstellen einfach realisieren lassen. Das Framework abstrahiert von den konkreten Sensoren weg, indem es Schnittstellen anbietet, mit denen sogenannte Production Nodes bedient werden können. Production Nodes implementieren verarbeitende Funktionalitäten, beispielsweise einen Menschendetektor in Tiefenbildern. Aufbauend auf einem oder mehrerer Production Nodes können wiederum andere Production Nodes ihre Datenverarbeitung ausführen. Die Kommunikation und der Datenfluss werden dabei vom OpenNI Framework reguliert. [11]

Die Anwendung, die sich der dabei erzeugten Daten bedient, muss zu Beginn ihrer Ausführung den Verarbeitungsgraphen erzeugen und in das Framework registrieren oder sie bedient sich einer bereits fest definierten Konfiguration. Dafür verwaltet das Framework eine Datenbank, in der global verfügbare Production Nodes hinterlegt sind. Feste Konfigurationen lassen sich mittels *XML*-Dateien festlegen, deren Pfade bei der Initialisierung des OpenNI Frameworks diesem übergeben werden können. In den *XML*-Dateien muss definiert sein, welche Production Nodes in der Anwendung genutzt werden und wie diese konfiguriert werden sollen. Um das oben genannte Beispiel zu erweitern, kann ein Menschendetektor, der auf Tiefenbildern aufbaut, seine Eingabedaten gespiegelt erhalten. Die Implementierung der Konfigurierbarkeit ist dabei Aufgabe der Production Nodes. [11]
Eine Beispielkonfiguration sieht so aus:

```
<OpenNI>
  <Licenses>
    <License vendor="PrimeSense" key="0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
  </Licenses>
  <Log writeToConsole="true" writeToFile="false">
    <LogLevel value="3"/>
    <Masks>
      <Mask name="ALL" on="true"/>
    </Masks>
    <Dumps>
    </Dumps>
  </Log>
  <ProductionNodes>
        <Node type="Image">
           <Configuration>
                <MapOutputMode xRes="640" yRes="480" FPS="30"/>
                <Mirror on="true"/>
            </Configuration>
        </Node>
    <Node type="Depth">
      <Configuration>
        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
        <Mirror on="true"/>
      </Configuration>
    </Node>
    <Node type="User" />
    <Node type="Scene" />

  </ProductionNodes>
</OpenNI>
```

Der Wurzelknoten muss der OpenNI-Knoten sein. Darunter kommen die globalen Konfigurationen, wie das Loggen, eventuell benötigte Lizenzen für die Sensoren, im Beispiel wird der Kinect-Sensor genutzt, für den ein Schlüssel benötigt wird und die Production Nodes. In diesem Beispiel werden vier Production Nodes angefordert: Farbbild (Image), Tiefenbild (Depth), Menschenerkennung (User) und Umgebung (Scene). In der Konfiguration muss nicht notwendigerweise darauf geachtet werden, wo die einzelnen Production Nodes ihre Daten herbeziehen. Während der Initialisierung der Knoten müssen diese beim Framework geeignete Eingabedatenquellen anfordern und entsprechend reagieren, wenn keine oder mehrere verfügbar sind. Kann kein Datenfluss zustande kommen, da ein Knoten keine Eingabedaten erlangen kann, so schlägt die Initialisierung des Frameworks fehl.
OpenNI wird unter der GPL und der LGPL veröffentlicht.

4.3.1 Architektur und Datenfluss

Zweck des OpenNI Frameworks ist es, von den eigentlichen Sensoren zu abstrahieren und Schnittstellen zu bieten, mit denen Erkenntnisse aus den jeweiligen Sensoren und Kombinationen gewonnen werden können. Dabei wird eine strikte Trennung von verarbeitenden und nutzenden Programmteilen angestrebt. Programmteile, die Daten aufbereiten, gehören in die Middleware, also in die Production Nodes. Programmteile, die nur Daten konsumieren, die von Production Nodes erzeugt wurden, sollten sich in der Endanwendung befinden, also in der obersten Schicht. In der Abbildung 8 wird dies

erläutert.

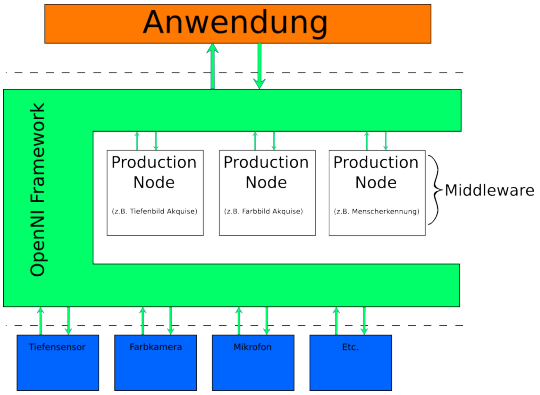


Abbildung 8: Architektur des OpenNI Frameworks [11]

Vorteilhaft an einer solchen Architektur ist, dass Production Nodes hochgradig wiederverwendbar sind und Verarbeitungsketten leicht erweitert oder angepasst weden können. Durch die Unterteilung der Verarbeitung in einzelne Module lassen sich diese gut organisiert parametrisieren. Die Kommunikation der durch die Production Nodes gewonnenen Daten geschieht ausschließlich über das Framework. Dadurch wird gewährleistet, dass jedes Modul, das bestimmte Typen von Daten erfordert, diese auch bekommt. Sind beispielsweise zwei Module auf eine Datenquelle registriert, bekommen beide bei Vorhandensein der Daten, also beim erfolgreichen Erzeugen dieser durch einen weiteren Production Node, diese nacheinander. Auf die Reihenfolge der Datenvermittlung hat man dabei keinen Einfluss. Daraus ergibt sich, dass Systeme auch schwingen können. Allerdings sollte beachtet werden, dass zwei Production Nodes, die einen gemeinsamen Eltern-Production Node haben, sich gegenseitig keine Informationen senden dürfen. Es darf also keinen Kreis in dem gerichteten Signalverarbeitungsgraphen geben [11].

4.3.2 NITE

Im Zuge der Entwicklung des OpenNI Frameworks hat PrimeSense einen Satz von Production Nodes bereitgestellt, der auf die Verwendung eines Tiefensensors (speziel der Kinect-Sensor) abzielt. In der NITE-Middleware sind Module enthalten, die die Nutzererkennung, die Skelettierung und die Gestenerkennung realisieren [14]. Für die meisten Programme, die die Funktionen des Kinect-Sensors nutzen, ist die NITE-Middleware ausreichend, um darauf basierend eigene Funktionen zu realisieren [7]. Im praktischen Teil dieser Arbeit wurde auch dieses Framework genutzt.

5 Steuerung durch Posen

Ein Ziel dieser Arbeit ist es die Bewegungen von fußballspielenden Robotern direkt zu steuern. Die Roboter sollen also die Posen des Nutzers so akkurat wie möglich einnehmen. Auf die dabei entstehenden Schwierigkeiten, soll im Folgenden näher eingegangen werden.

5.1 Winkel statt Positionen

Die NITE-Middleware stellt Mittel zur Verfügung, mit denen die räumlichen Positionen einzelner Körperteile berechnet werden können. Diese Punkte befinden sich in einem euklidischen Raum. Eine Schwierigkeit der Übertragung der Pose eines Nutzers besteht darin, die Orientierung des Nutzers so umzusetzen, dass die Roboter diese einnehmen können. Es wird also eine Funktion gesucht, die einen Vektor, der die Pose des Nutzers beschreibt, so umsetzt, dass sich die zugehörige Abbildung, so weit wie möglich, mit dem Raum der Vektoren, der Posen, die die Roboter einnehmen können, überschneidet. Eine komplette Abdeckung beider Räume ist nicht möglich, da den Robotern dafür notwendige Freiheitsgrade fehlen. Während Menschen an vielen Gelenken drei Freiheitsgrade (nicken, rollen, gieren) haben, haben die Roboter dort zwei oder nur einen.
Zur Lösung dieses Problems gibt es drei Ansätze:

- Positionsbasiert
  Die Orte der Gelenke werden durch eine Funktion, die die Ortsvektoren der einzelnen Gelenke in einen Raum umsetzt, der entsprechend der Robotergeometrie skaliert ist, umgesetzt und auf die Roboter übertragen.

  Problematisch an dieser Methode ist, dass die Roboter für jedes Gelenk einer kinematischen Kette, beispielsweise ein Arm, die Orte jedes Kindgelenkes in das Koordinatensystem des aktuellen Gelenkes umsetzen muss, damit Geometrieunterschiede zwischen dem Nutzer und den Robotern kompensiert werden können. Außerdem entsteht bei dieser Methode ein großer Rechenaufwand, da die Funktion, die die Punkte in das Roboterkoordinatensystem umsetzt für jeden Nutzer einzeln berechnet werden muss und nicht robust gegenüber Rauschen in der Skelettierung ist.

- Inverse Kinematik
  Es werden die Endeffektoren der Gliedmaßen, also die Hände und Füße

des Nutzers, zu den Robotern übertragen und dort mit einer inversen Kinematik in Posen umgesetzt. Die Endeffektoren müssen dabei wie bei der positionsbasierten Methode in ein Koordinatensystem umgesetzt werden, das die Robotergeometrie berücksichtigt. Die Roboter setzen eine Umkehroperation der Denavit-Hartenberg-Transformation ein, bei der der Lösungsraum sinnvoll eingeschränkt werden muss [13]. Auf den Robotern der FUmanoids passiert dies, indem die Gierachse in der Transformation weggelassen wird und die Winkel an den einzelnen Gelenken eingeschränkt werden. Damit existiert für jeden Endeffektor keine oder genau eine Lösung.

Die Probleme bei dieser Methode sind analog zur positionsbasierten Methode, jedoch entfällt die Schwierigkeit der Geometrieunterschiede für Teile einer kinematischen Kette. Bei der Nutzung einer inversen Kinematik kann es vorkommen, dass die Posen des Nutzers nicht wie gewünscht von den Robotern eingenommen werden, da die inverse Kinematik keine optimal nahe Lösung ergibt, die mit der Pose des Nutzers übereinstimmt. Winkt der Nutzer beispielsweise so, dass sein Oberarm senkrecht vom Oberkörper gestreckt ist und der Unterarm um den rechten Winkel nach oben schwingt, könnte eine inverse Kinematik diese Bewegung so auflösen, dass sich auch der Oberarm bewegt. Dieses Verhalten ist jedoch nicht erwünscht.

- Winkelbasiert
  Jedes Gelenk, das sowohl am Nutzer als auch am Roboter vorhanden ist, wird in Einzelgelenke aufgelöst. Damit wird das Nutzermodell an das Robotermodell angenähert. Für diese Einzelgelenke werden Winkel zwischen der aktuellen Konfiguration und einer Basiskonfiguration berechnet. Es gibt ein Modell der Basiskonfiguration für die Roboter, sodass diese die Winkel entsprechend umsetzen können. Schließlich nehmen sie eine Pose ein, die der des Nutzers ähnelt.

  Nachteile entstehen bei dieser Methode durch Rauschen der Skelettierung, das sich in kinematischen Ketten fortpflanzt. Kann beispielsweise die Position eines Knies nicht genau erfasst werden, so rauschen die Winkel am dazugehörigen Hüftgelenk übermäßig stark, womit die Positionen der Füße der Roboter noch stärker rauschen. Vorteilhaft ist dagegen, dass die Winkel auf dem Roboter sehr einfach umsetzbar sind, da die Servomotoren über ihre Sollwinkel angesprochen werden können.

Allen Methoden ist gemein, dass es gewisse Posen gibt, die nicht eindeutig bestimmbar oder nicht erreichbar sind. Streckt der Nutzer beispielsweise die Arme waagerecht aus (Abbildung 9), so gibt es für die Winkelberechnung

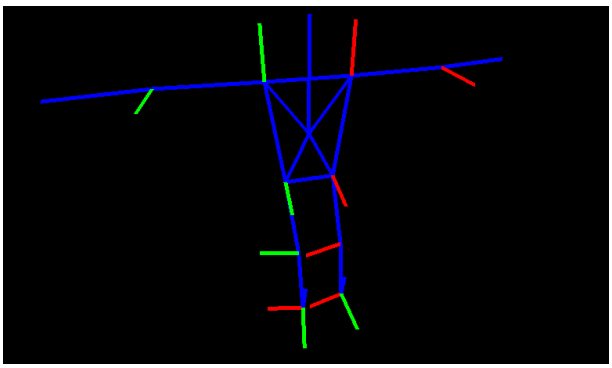


Abbildung 9: 3D Darstellung der Basispose mit den Normalenvektoren auf den Gelenken (rechts grün, links rot)

unendlich viele Lösungen im Nickgelenk des Roboters. Sind die Unterarme und die Oberarme nicht in der Ebene des Oberkörpers, können die Roboter diese Pose nicht nachempfinden, da ihnen ein Gelenk in den Schultern fehlt. Der Winkelbasierte Ansatz ist der effektivste, da hier die Berechnungen für alle Nutzer gleich funktionieren und keine Funktion genutzt werden muss, mit der Nutzer- in Roboterkoordinaten umgerechnet werden. So eine Funktion ist zu vermeiden, da sie nicht robust bei Menschen mit unterschiedlichen Körperproportionen ist. Des Weiteren ist anzumerken, dass die Funktion des OpenNI Frameworks, mit der für jedes Gelenk dessen Koordinatensystem bestimmt werden kann, nicht genutzt wird. Die von der Funktion gelieferten Koordinatensysteme sind auf Grund ihres Rauschens und ihrer Freiheitsgrade nicht geeignet, um damit die Winkel der Gelenke zu berechnen. Für die Basiskonfiguration wurde die Pose in Abbildung 9 gewählt.

5.2 Berechnung der Winkel der verschiedenen Gelenke

Die Roboter besitzen, im Gegensatz zu Menschen, keine Kugelgelenke. Es müssen also die meisten Gelenke des Nutzers in Repräsentationen umgesetzt werden, die denen der Roboter gleichen. Dabei werden in einigen Fällen Freiheitsgrade weggelassen, da diese an den Robotern nicht existieren.

Im Folgenden sollen die mathematischen Grundlagen erläutert werden, die für die daran anschließenden Berechnungen nötig sind.

5.2.1 Mathematische Grundlagen

- Skalarprodukt
  Das Skalarprodukt p zweier Vektoren $\vec{a}, \vec{b}$ ist definiert durch:

$$p = \sum_{i=1}^{dim(a)} a_i \cdot b_i \tag{1}$$

- Kreuzprodukt
  Das Kreuzprodukt $\vec{k}$ zweier Vektoren $\vec{a}, \vec{b} \in R^3$ ist definiert durch:

$$\begin{aligned} \vec{p} &= \vec{a} \times \vec{b} \\ p_1 &= a_2 \cdot b_3 - b_2 \cdot a_3 \\ p_2 &= a_3 \cdot b_1 - b_3 \cdot a_1 \\ p_3 &= a_1 \cdot b_2 - b_1 \cdot a_2 \end{aligned} \tag{2}$$

- Winkel zwischen Vektoren
  Ein Winkel zwischen zwei Vektoren $\vec{a}$, $\vec{b}$ berechnet sich folgendermaßen:

$$\alpha = \frac{\vec{a} * \vec{b}}{|a| * |b|} \tag{3}$$

  Diese Berechnung wird im Folgenden mit $\alpha = \vec{a} \angle \vec{b}$ abgekürzt.

  Ist dazu noch ein Vektor $\vec{c}$ gegeben, um den herum der Winkel gemessen werden soll (Rechte-Hand-Prinzip), so gilt:

$$\alpha = \begin{cases} \frac{\vec{a} \cdot \vec{b}}{|a| \cdot |b|} & \text{, für } \left(\vec{a} \times \vec{b}\right) \angle \vec{c} \leq 90° \\ \frac{\vec{a} \cdot \vec{b}}{|a| \cdot |b|} + 180° & \text{, für } \left(\vec{a} \times \vec{b}\right) \angle \vec{c} > 90° \end{cases} \tag{4}$$

  Dies wird im Folgenden mit $\alpha = \vec{a} \angle_{\vec{c}} \vec{b}$ abgekürzt.

- Matritzenmultiplikation
  Eine $a \times b$ Matrix A, die mit einer $b \times c$ Matrix B multipliziert wird, ergibt eine $a \times c$ Matrix C.

$$C_{i,j} = \sum_{k=1}^{b} A_{i,k} \cdot B_{k,j} \tag{5}$$

- Rotationsmatrix
  Rotationsmatritzen sind Matritzen mit der Vektoren um den Koordinatenursprung herum rotiert werden können. Eine Matrix A ist eine Rotationsmatrix, wenn gilt: $det(A) = 1$ und für jeden Vector $\vec{a}$ muss gelten $|A \cdot \vec{a}| = |\vec{a}|$. Eine Matrixin R^3, die um einen Vektor $\vec{v}$ mit dem Winkel α dreht bildet sich nach:
  Cosinus und Sinus werden mit c bzw. s abgekürzt.

$$A_{\vec{v},\alpha} = \begin{pmatrix} c(\alpha) + v_1^2\,(1 - c(\alpha)) & v_1 v_2\,(1 - c(\alpha)) - v_3 s(\alpha) & v_1 v_3 (1 - c(\alpha)) + v_2 s(\alpha) \\ v_1 v_2\,(1 - c(\alpha)) - v_3 s(\alpha) & c(\alpha) + v_2^2\,(1 - c(\alpha)) & v_2 v_3 (1 - c(\alpha)) + v_1 s(\alpha) \\ v_1 v_3 (1 - c(\alpha)) + v_2 s(\alpha) & v_2 v_3 (1 - c(\alpha)) + v_1 s(\alpha) & c(\alpha) + v_3^2\,(1 - c(\alpha)) \end{pmatrix} \tag{6}$$

- Orthogonale Projektionen

  Um einen Punkt P, beschrieben durch seine Koordinaten $\vec{p}$, auf eine Ebene in normierter Normalform $0 = (\vec{r} - \vec{a}) \cdot \vec{n}_0$ ($\vec{a}$ sei der Stützvektor, $\vec{n}_0$ der Normalenvektor) zu projizieren, werden folgende Operationen durchgeführt:

$$\vec{p}_{proj} = \vec{n}_0 \times (\vec{p} - \vec{a}) \times \vec{n}_0 + \vec{a} \tag{7}$$

- Invertieren von Rotationsmatritzen

  Die Inverse einer Rotationsmatrix ist ihre transponierte. Es gilt also:

$$E = A \cdot A^T = A \cdot A^{-1} \tag{8}$$

  [6] [8]

5.2.2 Modellierung

Ein Nutzer setzt sich für die Berechnungen aus den Koordinaten seiner Gelenke zusammen. In der Torsomitte befindet sich ein festes Gelenk, das als Körpermittelpunkt dient. In diesem befindet sich der Ursprung des Körperkoordinatensystems. Die x-Achse des Körperkoordinatensystems ist der Vektor vom Genick zur linken Schulter. Die y-Achse ist der Vektor zwischen Körpermittelpunkt und dem Genick und die z-Achse ist das Kreuzprodukt aus x- und y-Achse. Wie in Abbildung 10 sichtbar, garantiert die Skelettierung der NITE-Middleware, dass die Vektoren zwischen Körpermittelpunkt und Genick und zwischen Genick und den Schultern senkrecht aufeinander stehen.

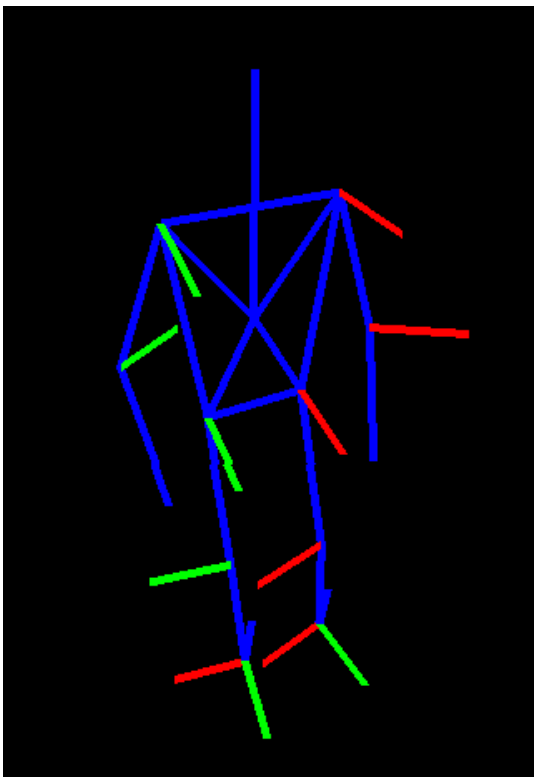


Abbildung 10: 3D-Darstellung eines Nutzers mit Normalenvektoren (links rot, rechts grün) auf den Gelenken und den Fußkoordinatensystemen

In Abbildung 10 ist die Skelettierung des Nutzers in einer 3D-Perspektive erkennbar. Die grünen und roten Linien an den Ellbogen-, Schulter-, Knie- bzw. Hüftgelenken stellen die errechneten Normalenvektoren auf den jeweiligen Gelenken dar. Grundsätzlich werden diese Normalenvektoren aus dem Kreuzprodukt des Vektors des Elternknochens mit dem des Kindknochens berechnet. In einigen Fällen ist dabei keine eindeutige Berechnung des Normalenvektors möglich. Diese Fälle werden im Folgenden erläutert.

- Ellbogen und Knie (Scharniergelenk)
  Der Winkel an einem Scharniergelenk am Punkt A, das Punkte B und C verbindet (siehe Abbildung 11) lässt sich durch

$$\alpha_{Scharniergelenk} = \frac{\vec{a} * \vec{b}}{|a| * |b|} \tag{9}$$

  berechnen. In der Implementierung wurde dabei eine Schwelle $\alpha_{ScharniergelenkMin}$ eingeführt, unterhalb der das Ergebnis 0 ist. Die vollständige Funktion

lautet also:

$$\alpha_{Scharniergelenk} = \begin{cases} \frac{\vec{a}*\vec{b}}{|a|*|b|} & \text{, für } \left|\frac{\vec{a}*\vec{b}}{|a|*|b|}\right| > \alpha_{ScharniergelenkMin} \\ 0 & \text{, sonst} \end{cases} \tag{10}$$

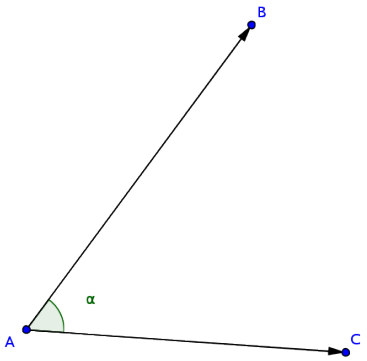


Abbildung 11: Winkel an Scharniergelenken

Die Normalenvektoren an diesen Gelenken werden mit dem Kreuzprodukt berechnet. Ist der Elternknochen fast parallel zum Kindknochen, das Gelenk ist nahezu um 180° gestreckt, muss der Normalenvektor anders berechnet werden. Das Vorhandensein dieses Vektors ist notwendig, um die Berechnung der korrekten Winkel an den Hüften und an den Schultern durchzuführen. Im Falle von annähernd parallelen Gliedmaßen werden folgende Berechnungen ausgeführt, um dennoch Normalenvektoren zu erhalten.

- Ellbogen
  Es wird eine Rotationsmatrix gebildet, von der die dritte Basis als Normalenvektor genommen wird.

$$\vec{n_{normal}} = M^{-1}_{Rotation} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{11}$$

  Die Rotationsmatrix wird, wenn möglich, aus dem Normalenvektor und dem Rollwinkel am Schultergelenk gebildet. Falls dies

nicht möglich ist, kann die y-Achse des Körperkoordinatensystems genutzt werden, da der Nutzer den Arm senkrecht vom Körper ausgestreckt hält. Hierbei wäre jeder Vektor der y-z-Ebene eine Lösung. Damit das Ergebnis eindeutig und so nah wie möglich an der Nullposition ist, wird die z-Basis genutzt (siehe 9) . Der Grundgedanke hinter der Nutzung der ersten Basis der Rotationsmatrix ist die Rotation der z-Achse um den Normalenvektor der Schulter.

– Knie
An den Knien wird analog zu den Ellbogen im Falle der Parallelität des Oberschenkels zum Unterschenkel eine Rotationsmatrix genutzt, mit der die x-Achse des Körperkoordinatensystems rotiert wird. Obwohl die Roboter drei Freiheitsgrade an diesen Gelenken haben, kann ein Freiheitsgrad vernachlässigt werden, da die Drehung des jeweiligen Beines um die eigene Achse nicht berechnet werden kann. Die dafür nötigen Informationen können aus den Orten der Gelenke nicht entnommen werden. Für die Erzeugung der Rotationsmatrix wird sich daher auf das Nick- und das Rollgelenk der Hüfte beschränkt. Die Winkel hierfür werden also berechnet, als hätte die Hüfte kein Giergelenk:

$$\alpha_{Huefte-Nick} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \angle \vec{v}_{Huefte-Knie} - 90° \tag{12}$$

$$\alpha_{Huefte-Roll} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \angle \vec{v}_{Huefte-Knie} - 90° \tag{13}$$

Aus den dadurch berechneten Winkeln lassen sich zwei Rotationsmatritzen erstellen, die multipliziert werden müssen.

$$M_{Rotation} = M_{Huefte-Roll} \cdot M_{Huefte-Nick} \tag{14}$$

$$\vec{n_{normal}} = M^{-1}_{Rotation} \cdot \vec{x} \tag{15}$$

- Schultern
Zur Berechnung der Gelenke an den Schultern muss der fehlende Freiheitsgrad berücksichtigt werden. Es müssen somit zwei Winkel gemessen werden. An den Robotern befinden sich zwischen Torso und Oberarm

die Nickgelenke und an diesen die Rollgelenke. Die Roboter können also den Oberarm nur dann um dessen Achse drehen, wenn dieser im rechten Winkel zum Oberkörper gestreckt ist. In diesem Fall ist das Nickgelenk gleichzeitig ein Giergelenk.

Für die Berechnung der Winkel an der Schulter werden bis zu drei Hilfsvektoren genutzt: die z-Achse des Körperkoordinatensystems, der Normalenvektor auf der Schulter und der Normalenvektor auf dem Ellbogengelenk. Der Normalenvektor an der Schulter wird mit dem Kreuzprodukt des Vektors vom Nacken zur Schulter und dem Vektor von der Schulter zum Ellbogen berechnet. Wenn diese Vektoren fast parallel sind, wird die y-Achse als Normalenvektor genutzt. Weiterhin muss die Orientierung des Normalenvektors eingeschränkt werden. Zeigt der Normalenvektor nach hinten-unten, so sind die sich daraus ergebenden Winkel nicht vom Roboter einnehmbar bzw. einzelne Kabel würden bei dieser Haltung unter mechanischer Spannung stehen und eventuell reißen. Um dies zu vermeiden, projiziert man den Normalenvektor orthogonal auf die y-z-Ebene und prüft, ob $\vec{v}_{projiziert} \angle \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \geq 135°$ gilt. Ist dies der Fall, wird der invertierte Normalenvektor genutzt. In Abbildung 12 werden die Vektoren im Modell gezeigt.

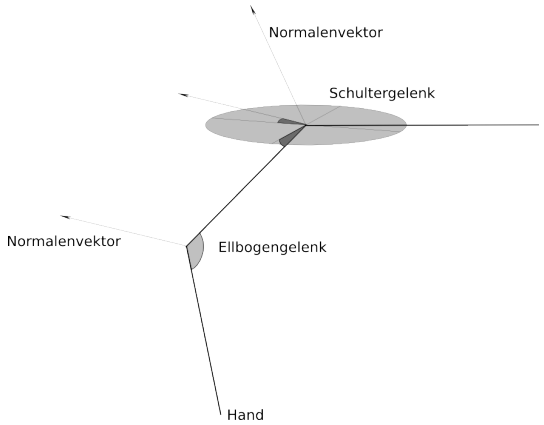


Abbildung 12: Winkel an der Schulter

Wie oben beschrieben, müssen zwei Winkel an den Schultern berechnet werden:

- Nickwinkel
  Der Nickwinkel ist der Winkel zwischen dem Nomalenvekor auf der Schulter und der y-Achse des Körperkoordinatensystems um die x-Achse desselben herum. Befindet sich der Oberarm nahezu parallel zum Schultergürtel, ist der Normalenvektor nur ungenau bestimmbar bzw. würde bei leichten Bewegungen starke Bewegungen an den Robotern verursachen. Hierfür muss der Normalenvektor am Schultergelenk in die Berechnung einbezogen werden. Der Nickwinkel ergibt sich dann aus dem Winkel zwischen dem Normalenvektor des Ellbogenegenks und der z-Achse des des Körperkoordinatensystems um die x-Achse herum.

- Rollwinkel
  Da die Fallunterscheidung für die Winkel bereits in der Nickwinkelberechnung getan wurde und die Roboter nur zwei physikalische Gelenke an den Schultern haben, berechnet sich der Rollwinkel aus dem Winkel zwischen dem Schultergürtel und dem Oberarm um den Normalenvektor der Schulter herum. In diesem Fall muss nicht unterschieden werden, ob der Oberarm annähernd parallel zum Schultergürtel ist, da angenommen werden kann, dass der Winkel 0° ist. Dies ergibt sich durch die normale Winkelberechnung.

- Hüfte

An den Robotern setzt sich das Hüftgelenk aus drei Servomotoren zusammen, die jeweils um eine Achse rotieren. In der kinematischen Kette kommt erst das Gier-, dann das Roll- und schließlich das Nickgelenk.

Wie schon bei der Schulter müssen auch an der Hüfte die Winkel einzeln berechnet werden. Siehe Abbildung 13.

- Gierwinkel
  Die Rotation um die y-Achse an diesem Gelenk wird zum Drehen der Roboter benötigt. Steht nur ein Bein auf dem Boden, kann durch Rotation an dieser Achse der Oberkörper gedreht werden. Geschieht dies abwechselnd mit beiden Beinen, dreht sich der

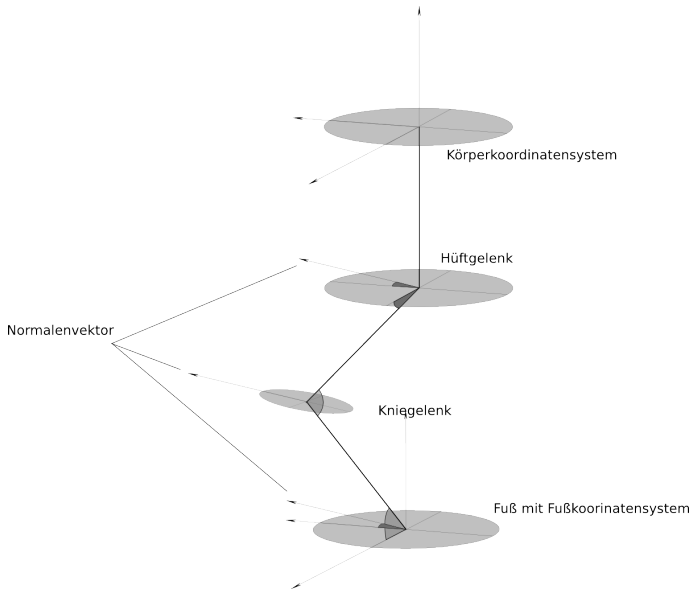


Abbildung 13: Winkel Berechnung an den Beinen

Roboter. Bei der Berechnung des Gierwinkels wird lediglich unterschieden, ob sich der zum Bein zugehörige Fuß auf dem Boden befindet oder nicht. Befindet er sich nicht auf dem Boden, ergibt sich der Gierwinkel mit dem Normalenvektor des Knies aus:

$$\alpha_{Gier} = (\vec{x}_{Basis} \angle \vec{n}_{Knie}) - 90° \tag{16}$$

Es wurden Koordinatensysteme eingeführt, die von den Füßen gehalten werden. Sie kommen zum Einsatz, wenn der Boden nicht von den jeweiligen Beinen berührt wird. Wenn der Fuß in der Luft ist, also den Boden nicht berührt, so ist das Koordinatensystem des Fußes das des Körpers M_K. Sobald der Fuß den Boden berührt und so lange er nicht wieder in der Luft ist, wird das letzte Koordinatensystem M_F gehalten. Dabei wird der Winkel folgendermaßen berechnet:

$$\vec{h} = \begin{pmatrix}0\\1\\0\end{pmatrix} \times \left(M_F^{-1} * \begin{pmatrix}1\\0\\0\end{pmatrix} \right) \times \begin{pmatrix}0\\1\\0\end{pmatrix}$$
$$\alpha_{Gier} = \vec{h}\angle \left(M_K^{-1} * \begin{pmatrix}1\\0\\0\end{pmatrix} \right) \tag{17}$$

Es wird somit ein Hilfsvektor erzeugt, mit dem die auf die x-z-Ebene projizierte x-Achse des Fußkoordinatensystems repräsentiert und der Winkel zwischen diesem Hilfsvektor und der x-Achse des Körperkoordinatensystems gemessen wird. Für bessere Ergebnisse kann dieser Vorgang ebenso mit den z-Achsen der einzelnen Koordinatensysteme durchgeführt werden. Dann wird überprüft, bei welcher der projizierte Hilfsvektor länger ist. Auf diese Art ist man robuster gegen den Fall, dass der Nutzer liegt. In der konkreten Implementierung wurde bewusst auf diesen Fall verzichtet.

– Rollwinkel
Der Rollwinkel ist der Winkel zwischen der x-z-Ebene und dem Normalenvektor des Knies sowie nach der Rechte-Hand-Regel um die zweite Basis einer Rotationsmatrix $M_{\vec{v},\alpha_{gier}}$ mit $\vec{v} = \begin{pmatrix}0\\1\\0\end{pmatrix}$:

$$\alpha_{Roll} = \left(\vec{y}\angle_{M^{-1}_{\vec{y},\alpha_{gier}3}} \vec{v}_{normal-Knie} \right) - 90° \tag{18}$$

Aus praktischen Gründen kann $M^{-1}_{\vec{y},\alpha_{gier}3} = \begin{pmatrix}0\\0\\1\end{pmatrix}$ angenommen werden. Die Berechnung ist dann korrekt, solange $-90° < \alpha_{Gier} < 90°$ gilt.

– Nickwinkel
Für die Berechnung des Nickwinkels wird wiederum eine Rotationsmatrix genutzt, mit der die z-Achse des Körperkoordinatensystems um die y-Achse mit dem Gierwinkel rotiert wird. Der Nickwinkel ergibt sich dann aus dem Winkel zwischen der rotierten z-Achse und dem Vektor des Oberschenkels um die zweite Basis von $M_{\vec{y},\alpha_{gier}}$ herum.

$$\alpha_{Nick} = \left(M^{-1}_{\vec{y},\alpha_{gier}} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \angle_{M^{-1}_{\vec{y},\alpha_{gier}1}} \vec{v}_{Huefte-Knie} \right) - 90° \qquad (19)$$

Die 90° ergeben sich daraus, dass der hier berechnete Winkel um 90° größer ist als der Nullwinkel in der Basispose.

- Füße

  Die Füße der Roboter haben zwei Gelenke, mit denen dem Nick- und Rollwinkel der Hüfte entgegengesteuert werden kann. Die Winkel an diesem Gelenk sollten nicht direkt berechnet werden, es sollten stattdessen die inversen Winkel der Hüfte genommen werden. Dadurch wird gewährleistet, dass die Roboter nicht umfallen.

5.3 Probleme durch unterschiedliche Körpergeometrien

Im Gegensatz zur Steuerung der Arme durch die oben beschriebenen Methoden, ist es nicht möglich die Beine der Roboter direkt zu steuern. Einerseits ist die Skelettierung des Nutzers zu rauschanfällig, andererseits stören die unterschiedlichen Geometrien und Gewichtsverteilungen zwischen Roboter und Nutzer. Die Roboter müssten sich selbstständig stabilisieren, um ihr Umfallen zu vermeiden, wenn die Beine gesteuert werden. Allerdings werden die Posen des Nutzers dann nur noch wenig umgesetzt, sodass der gewollte Nutzen, das direkte Fernsteuern des Roboters, nicht mehr sichtbar wäre. Weiterhin entstehen Schwierigkeiten in der Stabilisierung der Beine durch die Verschiedenartigkeit der Gelenke zwischen Menschen und den Robotern. Die Annahme, dass zum Beispiel Knie Scharniergelenke sind, ist nicht unbedingt zutreffend. Knie erlauben einen nicht zu vernachläsigbaren Spielraum für Gierbewegungen des Unterschenkels. Menschen nutzen diesen und weitere Freiheitsgrade ihrer Gelenke, um das eigene Laufen zu stabilisieren.

6 Steuerung durch Gesten

Da die Steuerung der Beine nicht ohne Weiteres realisierbar ist, müssen bestimmte Bewegungsabläufe (statische und dynamische Motions) anders realisiert werden. Beispielsweise ist es nicht gelungen die Roboter über die direkte Steuerung laufen oder aufstehen zu lassen. Wie oben beschrieben sind die Körpergeometrien und Gewichtsverteilungen zu unterschiedlich. Es

wurden für die notwendigsten Bewegungsabläufe der Roboter im Fußballspiel Konzepte ausgearbeitet, mit denen die Roboter dennoch von einem Menschen ferngesteuert werden könnnen, ohne das Konzept der natürlichen Schnittstellen zu verletzen. Diese werden im Folgenden erläutert. Bei der Steuerung durch Gesten werden nur Daten an die Roboter übertragen, die die Ausführung der Motions auf dem Roboter veranlassen und im Falle der Laufsteuerung noch die Parameter, die für das Laufen notwendig sind. Die konkrete Ausführung und die Berechnung, wie die Motion ausgeführt werden soll, findet ausschließlich auf den Robotern statt.
Als Gesten seien Bewegungsabläufe des Nutzers beschrieben. Sie sind unabhängig von der räumlichen Ausrichtung des Nutzers und sollten so gestaltet sein, dass sie für Nutzer intuitiv sind.
Im Folgenden muss zwischen statischen und dynamischen Motions unterschieden werden, da sie jeweils unterschiedliche Konzepte der Steuerung erfordern.

- Dynamische Motions
  Dynamische Motions sind beispielsweise das Laufen und das dynamische Schießen. Die Roboter können unterschiedlich schnell laufen und ihren Schuss entprechend der Spielsituation anpassen [12]. Aus der Sollgeschwindigkeit berechnet sich beispielsweise die Schrittweite. Die Ausführung der Bewegung wird also dynamisch berechnet.

- Statische Motions
  Bewegunsabläufe wie das Aufstehen von hinten oder vorn oder Bewegungen, die der Unterhaltung dienen, wie der Hühnertanz, laufen immer gleich ab. Ihre Ausführung dauert immer gleich lang und veranlasst die Roboter dazu, immer zum gleichen Zeitschritt innerhalb des Bewegungsablaufes die gleiche Konfiguration zu haben.

Da die meisten statischen Motions der Unterhaltung dienen und durch die direkte Steuerung abgedeckt werden können, müssen lediglich zwei statische Motions durch Gesten des Nutzers auslösbar sein können.

6.1 Funktionsweise

Die Gestenerkennung setzt auf der Skelettierung auf. Mittels Zustandsmaschinen werden die Konfiguration des Skelettes oder Teile dessen überwacht und der Zustand der Maschine verändert, wenn das Skelett eine bestimmte Pose eingenommen hat. Für die Steuerung von dynamischen Motions ist es sinnvoll, zusätzlich zu dem Zustand noch die Zeitpunkte zu speichern, an denen

die Zusandsübergänge stattgefunden haben. Dynamische Motions können so mit zusätzlichen Informationen beliefert werden. In diesem konkreten Fall wurde diese Anwendung nur bedingt genutzt, da außer der Laufsteuerung keine dynamischen Motions angesteuert werden.

6.2 Aufstehgeste

Eine intuitive Art den Roboter aufstehen zu lassen ist es, selbst aufzustehen. Der Nutzer muss also in die Hocke gehen und sich dann wieder aufrichten. Wichtig ist hierbei, dass eine gewisse vertikale Strecke von einem zentral gelengenen Körperteil überbrückt wird. Hierfür eignen sich die Position des Kopfes oder die des Körpermittelpunktes. Die Zustandsmaschine ist in Abbildung 14 dargestellt.

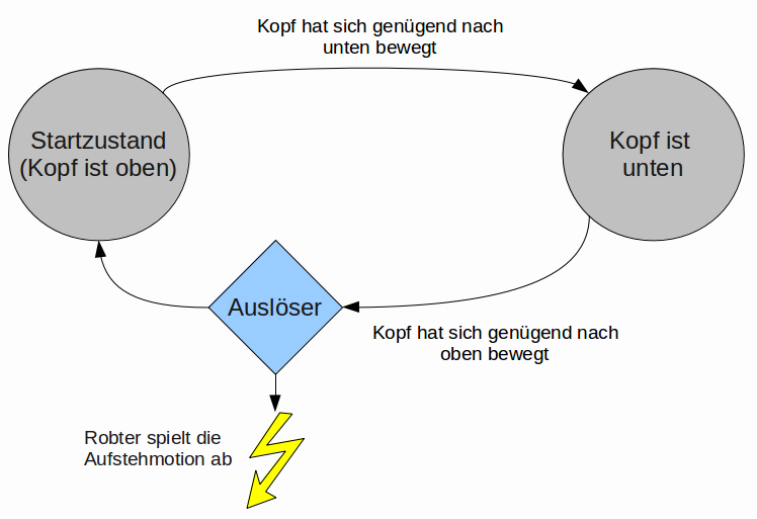


Abbildung 14: Zustandsdiagramm Aufstehgeste

Problematisch an dieser Zustandsmaschine ist, dass die Körperhöhe bekannt sein muss. Hierfür wurde eine Variable geführt, die den maximalen y Wert des Körperteils hält, der gemessen wird, während sich die Zustandsmaschine im Startzustand befindet. Ausgehend von diesem Wert und der Höhe des Bodens kann die Körperhöhe gemessen werden und wie tief sich ein Nutzer hocken muss, manit die Zustandsmaschine in den ”Kopf ist unten” Zustand wechselt.
Die Ausführung der Geste wird von den Robotern bestimmt. Stehen sie und es erreicht sie das Signal aufzustehen, so wird das Signal verworfen. Sie unterscheiden auch selbstständig, welche Aufstehbewegung angewendet werden soll, je nachdem ob sie auf dem Rücken oder auf dem Bauch liegen.

6.3 Schussgeste

Die Schussbewegung der Roboter ist der eines Menschen stark nachempfunden. Sie verlagern das eigene Gewicht auf das Stützbein und bewegen das Schussbein nach hinten, um dieses danach nach vorn schnellen zu lassen. Prinzipiell können die Roboter mit beiden Beinen schießen [12]. Die Steuergeste für den Schuss ist daher eine Schussbewegung des Menschen. Sollen die Roboter mit dem linken Bein schießen, so muss der Nutzer die Schussgeste mit dem linken Bein ausführen und andersherum vice versa. Die Zustandsmaschine muss in diesem Fall noch berücksichtigen, dass entweder eine Schussgeste mit dem linken Bein gewünscht ist oder mit dem rechten. Ist sie also in einem Zustand in dem die Schussgeste eines Beines initialisiert ist (das Bein wurde nach hinten geschwungen), so muss es eine Möglichkeit geben, wieder in den Initialzustand gelangen, ohne den Schuss auszuführen. Dafür wurde ein Timer genutzt. Ab dem Zeitpunkt, an dem der Nutzer das Schussbein nach hinten genommen hat, hat er zwei Sekunden Zeit, um die Geste auszuführen. Verstreichen diese, ohne dass er sein Bein nach vorn geschwungen hat, gelangt die Zustandsmaschine automatisch wieder in den Startzustand. Hält er dagegen diese Position (das Bein ist also länger als zwei Sekunden nach hinten gestreckt), gelangt die Maschine automatisch wieder in den Initialisierungszustand der Schussgeste für das entsprechende Bein. Wartet der Nutzer also auf einen bestimmten Zeitpunkt für den Schuss, kann er sein Schussbein hinten lassen und erst dann nach vorn schnellen lassen, wenn der Schuss ausgeführt werden soll. Die Zustandsmaschine wird in Abbildung 15 dargestellt.

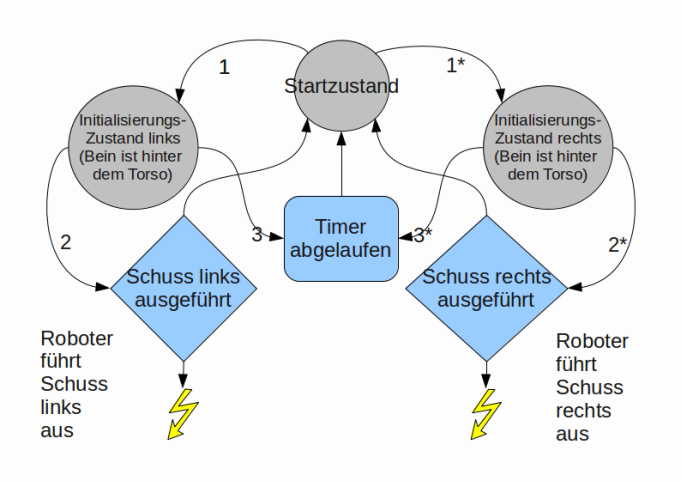


Abbildung 15: Zustandsdiagramm der Schussbewegung

Da der Automat symmetrisch für das linke und rechte Bein ist, die Überführungen aber das gleiche bedeuten, wird zwischen den Überführungen un-

terschieden, indem die Funktionen mit * für das rechte Bein gelten. Die Überführungsfunktionen bedeuten das Folgende:

- 1: Der Nutzer hat sein Bein nach hinten gestreckt. Damit wird in den Initialisierungszustand für das entsprechende Bein gewechselt.
- 2: Der Nutzer hat das Bein nach vorn geschwungen. Damit wird das Signal an die Roboter geschickt, dass der Schuss am entsprechenden Bein ausgeführt werden soll.
- 3: Der Nutzer hatte sein Bein nach hinten gestreckt, jedoch nicht im entsprechenden Zeitfenster nach vorn geschwungen. Damit wird wieder in den Ausgangszustand gewechselt.

Für die Grundlage der Überführungsfunktionen hat sich herausgestellt, dass es nicht zielführend ist, die Winkel zwischen den jeweiligen Gelenken zu betrachten. In der Praxis starten viele Nutzer ihre Schussphase mit dem nach vorn Kippen des Oberkörpers. Dabei lassen sie ihr Schussbein gestreckt. Wird der Nickwinkel am Hüftgelenk von der Zustandsmaschine beobachtet, kann keine Änderung erkannt werden, da sich die Orientierung des Schussbeines zum Oberkörper nicht geändet hat. Auch ist es nicht zielführend, wenn man stattdessen den Nickwinkel des anderen Beines beobachtet. In diesem Fall wird kein Schuss erkannt, wenn der Nutzer seinen Oberkörper aufrecht hält und nur das Schussbein schwingen lässt. Als praktisch hat sich die horizontale Entfernung zwischen Körpermittelpunkt und Knie egeben. Überschreitet diese einen gewissen Schwellwert und befindet sich das Schussbein nicht auf dem Boden, wird in den Initialisierungszustand übergegangen. Das Erkennen der weiteren Phasen der Geste erfolgt analog.

7 Laufsteuerung

Das Laufen ist eine dynamische Bewegung, deren genaue Ausführung auf dem jeweiligen Roboter berechnet werden muss. Der Roboter berechnet nach seiner Geometrie die Schrittweite, die Drehungen der Gier- und Rollgelenke der Hüfte bzw. Füße. Das FUmanoid-Programm stellt hierfür ein Objekt zur Verfügung, das eine komfortable Laufsteuerung realisiert. Mit dem Objekt lässt sich die Vorwärts-, Rotations- und Seitwärtsgeschwindigkeit ändern, ohne dass der genaue Laufmechanismus bekannt ist. Auf der Seite des Nutzers müssen Werte generiert werden, die den Sollgeschwindigkeiten der Roboter entsprechen. Zur Berechnung wird ein virtueller Kreis zur Hilfe genommen, der im Nutzerraum horizontal fest um einen Punkt liegt. Dieser Kreis kann

entweder global fest sein oder für die einzelnen Nutzer selbst berechnet werden. Befindet sich der Nutzer nah dem Mittelpunkt, bleibt der Läufer stehen. Stellt sich der Nutzer nun weiter vorn in den Kreis, läuft der Roboter vorwärts bzw. seitlich, wenn sich der Nutzer nach links oder rechts stellt. In Abbildung 16 ist dieser Kreis dargestellt. Hat sich der Nutzer um die vertikale Achse gedreht, so dreht sich der Roboter entsprechend. Die Geschwindigkeiten, mit denen der Roboter die Bewegungen ausführt richten sich bei der Laufrichtung nach der räumlichen Entfernung zwischen Nutzer und Mittelpunkt des Kreises und bei der Rotation am Winkel zwischen einer Nullrichtung und der aktuell gemessenen Richtung. Die Nullrichtung kann wie der Mittelpunkt des Kreises global oder für jeden Nutzer einzeln festgelegt werden.

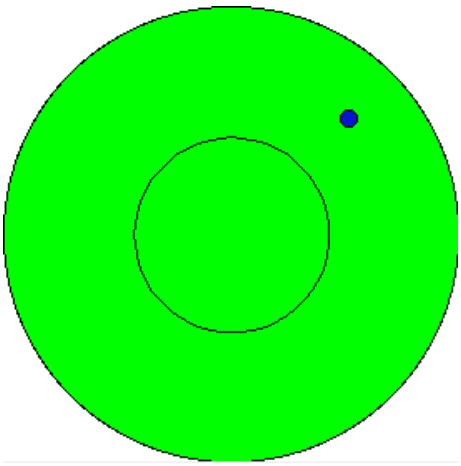


Abbildung 16: Visualisierung der Laufsteuerung. Der Roboter wird vorwärts und leicht nach rechts gesteuert.

In Bild 16 wird die Laufsteurung visualisiert. Der rote Punkt stellt die Körpermitte des Nutzers dar, der sich innerhalb des virtuellen Kreises bewegt. Der innere Kreis stellt eine Schwelle dar. Befindet er sich außerhalb des inneren Kreises, fangen die Roboter an, sich in die entsprechende Richtung zu bewegen. Verlässt der Nutzer den äußeren Kreis, so bleibt der Roboter stehen. Er lässt sich dann erst wieder steuern, wenn der Nutzer wieder den inneren Kreis betreten hat.

8 Fazit

Wie bereits erläutert funktioniert die Laufsteuerung nicht mit der direkten Steuerung. Wünschenswert wäre es, wenn die Roboter die Beinposen so genau wie möglich einnehmen könnten, die der Nutzer vorgibt. Derzeit ist dies aber

nicht möglich, da die Skelettierung die Beine nicht genau genug auflösen kann. Eine Lösung an dieser Stelle wären Markierungen an den Beinen, mit denen diese besser erkannt werden können. Dann bestünde dennoch das Problem der unterschiedlichen Geometrien. Die Roboter fallen bei bestimmten Posen um, bei denen Menschen stabil stehen und andersherum. Es muss also eine Schnittmenge dieser Posen gefunden werden und nur Posen aus dieser Schnittmenge übertragen werden. Allerdings ist die Schnittmenge nicht trivial zu berechnen, da einerseits die genauen Gewichtsverteilungen der Roboter nicht bekannt und die der Nutzer nur kompliziert zu messen sind. Für weitere bessere Ergebnisse müssen einige Freiheitsgrade der Menschen wegabstrahiert werden. Die Gierbewegungen der Unterschenkel beispielsweise sind nicht von den Robotern reproduzierbar.
Der praktische Teil dieser Arbeit wurde im Rahmen der langen Nacht der Wisschenschaften an der Freien Universität präsentiert und fand speziell bei dem jüngeren Publikum großen Anklang. Die Kinder konnten dabei aus der Roboterperspektive die Arme steuern, zum Ball navigieren und Tore schießen. Die Steuerung durch Gesten schien den Kindern besonders leicht gefallen zu sein. Auch ohne vorherige Erklärung der Steuerung waren die Kinder in der Lage die richtige Schuss- und Aufstehgeste auszuführen.

9 Ausblick

Natürliche Schnittstellen werden sich in den kommenden Jahren weiter im Alltag etablieren. Derzeit gibt es bereits einige vielversprechende Ansätze, um Eingaben für Computer zu generieren. Diese sind allerdings nur auf spezielle Anwendungen zugeschnitten. Langfristig werden sich daher wahrscheinlich Kombinationen aus Bedienkonzepten durchsetzen, sodass Nutzer auf die Art mit ihren Computern interagieren können, wie sie es bevorzugen.
Im Bereich Robotik kann mit natürlichen Schnittstellen besser auf Menschen reagiert werden. Sie bieten speziell für ältere Menschen eine gute Möglichkeit der Mensch-Maschinen-Interaktion, bei denen das Einlernen in klassische (nicht-natürliche) Bedienkonzepte schwer ist. Haushaltsroboter können zum Beispiel durch Gesten darauf aufmerksam gemacht weden, bestimmte Dinge zu erledigen.
Steuerungen wie sie im Rahmen dieser Arbeit realisiert wurden, können in Bereichen genutzt werden, wo Menschen nicht arbeiten können, weil die Umgebung der Gesundheit schaden würde. Roboter, können darin Arbeit verrichten, für die menschliche Geschicklichkeit und schnelle Anpassungsfähigkeit notwendig ist. So können komplizierte Reperaturaufgaben von Experten durchgeführt werden, ohne dass diese anwesend sein müssen.

Literatur

[1] Wikipedia: Deep map from Kinect, verfügbar online unter: `http://upload.wikimedia.org/wikipedia/commons/9/90/Kinect2-deepmap.png`.

[2] Wikipedia: Von der Kinect empfangenes Infrarotbild mit dem vom Laserprojektor ausgestrahltem Punktmuster, verfügbar online unter: `http://upload.wikimedia.org/wikipedia/commons/7/76/Kinect2-ir-image.png`.

[3] Boguslaw Cyganek and J. Paul Siebert. *An Introduction to 3D Computer Vision Techniques and Algorithms*. Wiley & Sins, Ltd, 2009.

[4] Oliver Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint*. The MIT Press, 1993.

[5] Adrian Kaehler Gary Bradski. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008.

[6] John H. Hubbard and Barbara Burke Hubbard. *Vector Calculus, Linear Alegebra, and Differential Forms*. Martrix Editions, 2009.

[7] jkj. Minority Report im Fernsehsessel. *C'T*, 11:168 – 171, Mai 2011.

[8] Marvin Marcus and Henryk Minc. *Integrated Analytic Geometry and Algebra with Circular Functions*. General Learning Press, 1973.

[9] Dave Mark and Jeff LaMarche. *Beginning Iphone 3 Development: Exploring the Iphone SDK*. Apress, 2009.

[10] Hans-Arthur Marsiske. RoboCup German Open: "Roboter werden Überschätzt". *Heise Online*, 2011.

[11] Natural Interaction. *OpenNI User Guide*.

[12] Stefan Otte. Entwicklung eines dynamischen Schusses für humanoide Fußballroboter, 2010.

[13] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Controll - The Computer Control of Robot Manipulators*. The MIT Press, 1981.

[14] PrimeSense. *Prime Sensor^{TM}NITE 1.3 Algorithms notes*, 2010.

[15] PrimeSense. *PrimeSense Reference Design 1.08PS*, 2010.

[16] Robotis Inc. *User's Manual Dynamixel RX-28.*

[17] Robotis Inc. *User's Manual Dynamixel RX-64.*

[18] Daniel Seifert, Hamid Reza Moballegh, Prof. Raúl Rojas, and et al. FUmanoids Humanoid League - KidSize Team Description Paper 2011.

[19] Emanuele Trucco and Alessandro Verri. *Indroductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.